



Tema 2. Algoritmi: caracteristici, reprezentare, implementare

Obiective

- să urmărești execuția algoritmilor pas cu pas
- să identifici valorile variabilelor la fiecare pas
- să creezi algoritmi repetitivi

Fișa de documentare 2.3. Programarea structurată (III)



Structura repetitivă

O structură repetitivă se caracterizează prin posibilitatea efectuării repetitive a unei secvențe de instrucțiuni, **cât timp** este îndeplinită o anumită condiție sau **pâna când** se îndeplinește o anumită condiție. Repetiția secvenței de instrucțiuni se numește „**iterație**”.

Structurile repetitive se mai întâlnesc sub numele de structuri ciclice sau cicluri.

Există trei tipuri de structuri repetitive:

- Structura cu număr necunoscut de repetiții cu test inițial (**CÂT TIMP** sau **WHILE**)
- Structura cu număr necunoscut de repetiții cu test final (**EXECUTA - CÂT TIMP** sau **DO-WHILE**)
- Structura cu număr cunoscut de repetiții (**PENTRU** sau **FOR**)



Structura repetitivă cu test inițial - **CÂT TIMP** sau **WHILE**

Structura repetitivă cu test inițial are două componente:

- **conditia**, o expresie logică ce poate fi evaluată prin valoarea TRUE sau FALSE, condiție pe care o notăm cu **c**;
- **actiune**, o secvență de instrucțiuni ce se vor executa repetat, notată cu **a**, acțiune asociată cu EXECUTĂ;

Folosind notațiile făcute, structură repetitivă cu test inițial se poate scrie astfel:

```
cât timp c execută
|   a
|
```



Principiul de executare este următorul:

Cât timp condiția **c** este adevărată, se execută secvența de instrucțiuni „a”. Execuția se oprește când condiția **c** nu mai este adevărată.



Observații:



Pentru ca structură repetitivă să nu intre într-un ciclu infinit, trebuie ca secvența de instrucțiuni să modifice cel puțin una din variabilele care intervin în condiție astfel încât aceasta să poată deveni falsă la un moment dat.



Dacă de la bun început condiția are valoarea **fals**, secvența de instrucțiuni **nu se execută nici măcar o dată**.



Exemplu: Suma cifrelor

Să considerăm următoarea problemă: **Se citește un număr n , întreg, cu cel mult 9 cifre. Se cere să se afișeze suma cifrelor numărului n .**

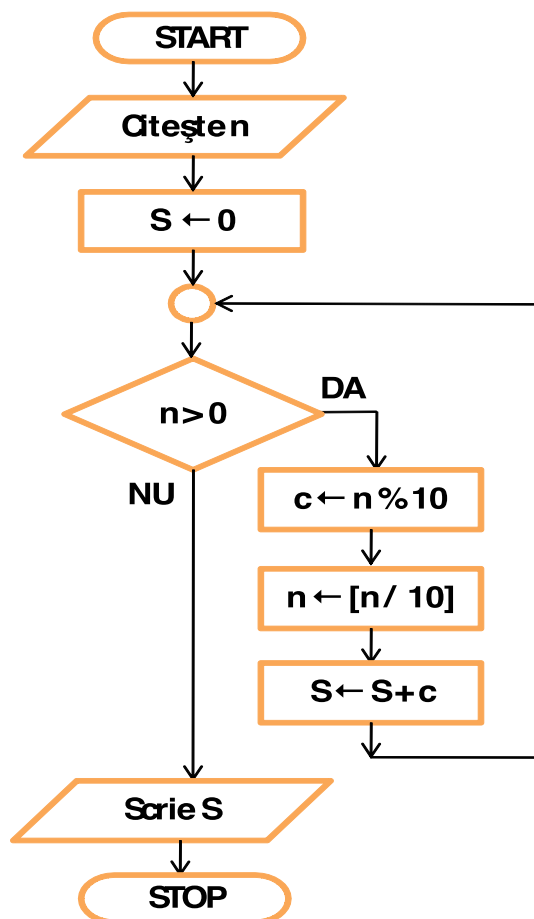


Explicarea algoritmului: Rezolvarea presupune că se extrag pe rând cifre din număr și se adaugă la sumă. Soluția se poate exprima în cuvinte astfel: **CAT TIMP** numărul este diferit de zero (deci mai sunt cifre de extras), **EXECUTĂ** extrage și elimină ultima cifră din numărul n apoi adaugă cifra la sumă.

Soluția are două componente:

- **condiția**, transcrisă prin “ numărul este diferit de zero”;
- **acțiune** transcrisă prin “ extrage și elimină ultima cifră din numărul n apoi adaugă cifra la sumă”;

Notând numărul dat cu „ n ”, cifra eliminată cu „ c ” și suma cifrelor cu „ s ”, algoritmul de rezolvare va fi:



```

n întreg //date de intrare
c întreg //date de manevră
S întreg //date de ieșire

```

```

citește n
S ← 0
cât timp n > 0 execută
|   c ← n % 10 //extrag cifra
|   n ← n / 10 //elimin cifra
|   S ← S + c //adun la sumă
|   ■
scrie s

```



Structura repetitivă cu test final – EXECUTĂ – CÂT TIMP sau DO - WHILE

Ca și structură repetitivă cu test inițial, structură repetitivă cu test final are aceleași două componente:

- **condiția**, o expresie logică ce poate fi evaluată prin valoarea TRUE sau FALSE, condiție pe care o notăm cu **c**;
- **acțiune**, o secvență de instrucțiuni ce se vor executa repetat, notată cu **a**, acțiune asociată cu EXECUTĂ;



În structură repetitivă cu test final mai întâi se execută secvența de instrucțiuni “a” și apoi se evaluează condiția. De aici și numele de structură cu test final.

În pseudocod forma generală a structurii repetitive cu test final este:

```

execută
|   a
|   cât timp c

```



Denumim această structură în mod obișnuit EXECUTĂ – CÂT TIMP sau DO – WHILE, însă există variante la fel de utilizate ale acestei structuri și anume: REPETĂ – PÂNĂ CÂND sau REPEAT – UNTIL.



Observații:



Pentru ca structură repetitivă să nu intre într-un ciclu infinit, trebuie ca secvența de instrucțiuni să modifice cel puțin una din variabilele care intervin în condiție astfel încât aceasta să poată deveni falsă la un moment dat.



Spre deosebire de structură repetitivă cu test inițial, structură repetitivă cu test final **efectuează o dată secvența de instrucțiuni** înainte de a testa condiția.



Exemplu: numărul de vocale

Să considerăm următoarea problemă: **Se citește de la tastatură o propoziție scrisă cu litere mici, terminată cu , . ' (punct) . Se cere să se afișeze numărul de vocale din propoziție.**



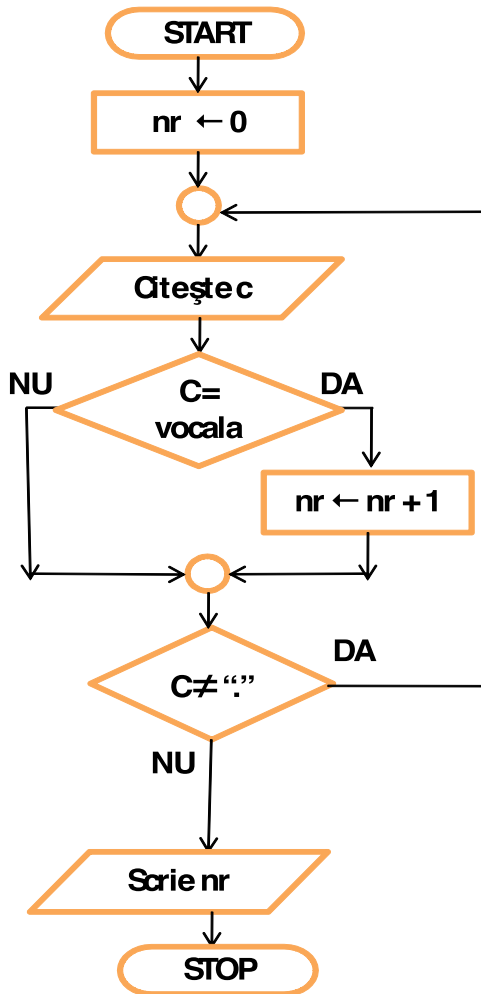
Explicarea algoritmului: Rezolvarea problemei se face citind succesiv caracterele din propoziție în variabila „c”, până citim caracterul , . '. Pentru fiecare caracter citit verificăm dacă este vocală (a, e, i, o, u) și dacă da, îl numărăm.

Soluția se poate exprima în cuvinte astfel: **EXECUTĂ** citește un caracter și dacă este vocală se crește contorul **CÂT TIMP** caracterul citit este diferit de , . '.

Avem două componente:

- **condiție**, transcrisă prin “ **caracterul citit este diferit de punct** ”;
- **acțiune** transcrisă prin “ **citește un caracter și dacă este vocală se crește contorul**”;

Rezolvarea algoritmului din exemplu este:



```

c caracter //date de intrare
nr întreg //date de ieșire

nr ← 0
execută
| citește c
| dacă (c='a' sau c='e' sau
|   c='i' sau c='o' sau c='u')
|   atunci
|   nr ← nr + 1 //numar vocala
|   ■
|   cât timp c≠"."
scrie nr
  
```



Structura repetitivă cu număr cunoscut de repetiții – PENTRU sau

FOR

În pseudocod forma generală a structurii repetitive cu număr cunoscut de repetiții este:

```

pentru contor ← exp_i, exp_f execută
|   a
|   ■
  
```

unde:

- **exp_i** și **exp_f** sunt expresii ale căror valori sunt evaluate în cadrul repetițiilor;
- **contor** este o variabilă ce va lua prima dată valoarea expresiei inițiale **exp_i**, urmând apoi să se modifice până la valoarea expresiei finale **exp_f**;
- **a** este secvență de instrucțiuni ce se va executa repetat;



Principiul de funcționare al structurii repetitive cu număr cunoscut de repetiții este următorul (am făcut presupunerea că **exp_i** ≤ **exp_f**):



Pasul 1: Se evaluează **exp_i** (expresia inițială);

Pasul 2: Se atribuie variabilei **contor** valoarea expresiei **exp_i**;

Pasul 3: Se evaluează **exp_f** (expresia finală);

Pasul 4: Dacă valoarea variabilei **contor** este mai mare decât valoarea expresiei **exp_f**, atunci se iese din structură repetitivă. Dacă valoarea variabilei **contor** este mai mică sau egală cu valoarea expresiei **exp_f**, atunci se execută secvența de instrucțiuni „a” și se incrementează (își mărește valoarea cu 1) valoarea variabilei **contor**, după care se reia pasul 3.



Observații:



Exp_i și exp_f pot fi expresii de evaluat sau doar variabile simple ce au valori date.



De regulă folosim structuri repetitive cu număr cunoscut de repetiții în care dorim ca **variabila contor să crească** de la **exp_i** la **exp_f**, caz în care evident valoarea **exp_i** trebuie să fie mai mică decât **exp_f**.

Într-o astfel de structură, secvența de instrucțiuni se execută de (**exp_f – exp_i + 1**) ori. Dacă însă folosind acest tip de structură, valoarea inițială a lui **exp_i** este mai mare decât **exp_f**, atunci secvența de instrucțiuni „a” nu se execută niciodată.

Dacă forma structurii **PENTRU** este:

```

pentru contor ← exp_i, exp_f, x execută
|   a
| ■

```

unde x este o variabilă numerică, atunci contorul va crește din x în x. Când x lipsește din structură **PENTRU**, contorul, crește cu 1.



În structurile repetitive cu număr cunoscut de repetiții în care dorim ca **variabila contor să scadă** de la **exp_i** la **exp_f**, trebuie ca **exp_i >= exp_f**, iar variabila contor va scădea din x în x sau cu câte o unitate. Dacă însă folosind acest tip de structură, valoarea inițială a lui **exp_i** este mai mică decât **exp_f**, atunci secvența de instrucțiuni „a” nu se execută niciodată.



Exemplu: Suma primelor n numere naturale



Se consideră următoarea problemă: Se citește un număr natural n . Să se afișeze suma primelor n numere naturale.

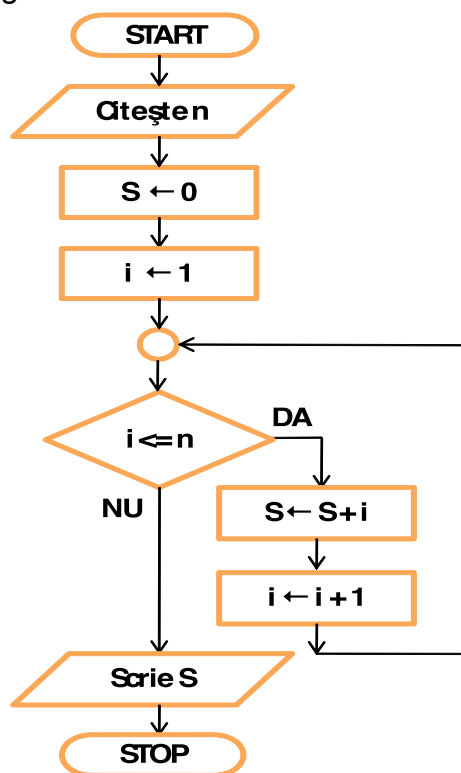
De exemplu dacă $n = 10$ atunci algoritmul va afișa 55, deoarece

$$S = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55.$$



Explicarea algoritmului: Se observă ca trebuie să calculăm o sumă cu număr cunoscut de termeni. Rezolvarea problemei se face deci folosind o structură repetitivă cu număr cunoscut de repetiții.

Algoritmul va fi următorul:



```

n întreg //date de intrare
S întreg //date de ieșire
i întreg //date de manevră
  
```

```

citește n
S ← 0
i ← 1
  
```

```

pentru i = 1, n execută
| S ← S + i
| ■
scrie S
  
```



Se observă că folosind scheme logice nu se poate reprezenta structură PENTRU decât cu ajutorul uneia din structurile repetitive cu testare inițială sau finală.



Transformări dintr-un tip de structură repetitivă în altul



Simularea structurii repetitive **WHILE** cu **DO-WHILE** se face astfel:

<pre> cât timp c execută a █ </pre>	<pre> Dacă c atunci execută a cât timp c █ </pre>
---	---



Se observă că este necesară testarea inițială a condiției deoarece, spre deosebire de structură **WHILE**, structură **DO-WHILE** efectuează cel puțin o dată secvența înainte de a testa condiția.



Simularea structurii repetitive **DO-WHILE** cu **WHILE** se face astfel:

<pre> execută a Lcât timp c </pre>	<pre> a //se execută secvența a cât timp c execută a █ </pre>
--	---



Se observă că în acest caz este necesar să executăm o dată secvența de instrucțiuni în afara ciclului.

Cele două structuri (**WHILE** și **DO - WHILE**) sunt echivalente (nefiind necesară existența ambelor), însă în funcție de problemă, vom alege structură repetitivă adecvată, care este mai potrivită pentru descrierea clară a algoritmului.



Simularea structurii repetitive **FOR** cu **WHILE** se face astfel:

<pre> pentru contor ← vi,vf execută a; █ </pre>	<pre> Contor ← vi cât timp contor<=vf execută a; contor ← contor + 1; █ </pre>
---	---



Se observă că în acest caz este necesar să scriem explicit instrucțiunea care crește contorul cu 1.



Simularea structurii repetitive **FOR** cu **DO - WHILE** se face astfel:



```

┌ pentru contor ← vi, vf execută
│   a;
│   ┌ Dacă contor ≤ vf atunci
│   │   ┌ execută
│   │   │   a;
│   │   │   contor ← contor + 1;
│   │   └ cât timp contor ≤ vf
│   └
└
```



Dintre toate aceste structuri repetitive, singura indispensabilă este cea cu test inițial (CÂT TIMP sau WHILE), celelalte putând fi obținute din aceasta, după cum am văzut mai sus.